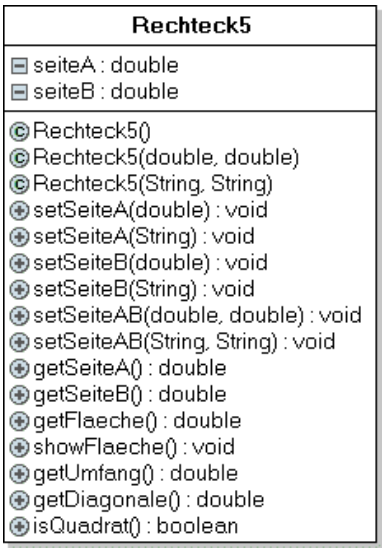
### Aufgabe 1: Einfache Klassendefinition

Erstellen Sie eine öffentliche Klasse **Rechteck5** mit den gekapselten Eigenschaften *seiteA* und *seiteB* und diversen Methoden (vgl. UML-Diagramm):




- Eingabemethoden s*etSeiteA, setSeiteB* bzw. *setSeiteAB*: Diese Methoden sollen die Seitenlängen als Parameter (*double*-Werte) erhalten. Bei nicht plausiblen Längenangaben soll der Wert der Attribute immer auf 0 gesetzt werden.

- Zur Ausgabe dienen die Methoden *getSeiteA* bzw. *getSeiteB*. Diese Methoden haben keinen Übergabeparameter, liefern jedoch die jeweilige Seitenlänge zurück.

- Die Eingabemethoden *setSeiteA, setSeiteB* bzw. *setSeiteAB* sollen überladen werden; d.h. entweder mit double- oder mit String-Parametern funktionsfähig sein.

- Mehrere Konstruktoren: Der Standardkonstruktor soll die Seitenlängen auf 0 setzen. Die überladenen Konstruktoren sollen nach Möglichkeit die bereits implementierten Methoden benutzen.

- *getFlaeche()* berechnet den Flächeninhalt des Rechtecks.

- *showFlaeche()* berechnet den Flächeninhalt und gibt ihn am Bildschirm aus.

- *getUmfang()* berechnet den Umfang des Rechtecks.

- *getDiagonale()* berechnet die Länge der Diagonalen.

- *isQuadrat()* gibt an, ob das Rechteck ein Quadrat ist.

- Die Klasse *Rechteck5* soll zusätzlich eine statische Variable (Klassenvariable) besitzen, die die Anzahl der erzeugten Rechtecke (=Anzahl der Objekte) enthält. Die statische Methode getAnzahlRechteck() soll die aktuelle Anzahl zurückgeben.

### Aufgabe 2: Rechnerverwaltung

Es soll mit einem Java-Programm eine einfache Rechnerverwaltung realisiert werden. Mit dem Programm werden exemplarisch 3 Rechner verwaltet.

Für einen Rechner sind folgende Daten zu speichern:

- Computername (Computername, max. 15 Zeichen, z.B. „asterix“)
- MAC-Adresse(n)
- Domäne (z.B. „bmw.de“
- IP-Adresse (z.B. „10.3.102.12“)
- Standort
- Usw.
- Die Daten sollen gekapselt werden.

Ein Rechnerobjekt muss über folgende Fähigkeiten verfügen:

- Automatische Initialisierung der Attribute mit den realen Daten eines Rechners bei der Objektinstanzierung.
- Ausgabe der Daten eines Rechners auf dem Bildschirm.
- Rückgabe einzelner Attribute an das aufrufende Modul (Getter-Methoden)
- Veränderung/Setzung eines einzelnen Attributes (Setter-Methoden).

2.1 Entwerfen Sie die notwendige Klasse und stellen Sie das Ergebnis in einem UML-Klassendiagramm dar.

2.2 Codieren Sie den Algorithmus in Java. Testen Sie das Programm.

### Aufgabe 3: Interpretation von Java-Quelltext

Analysieren Sie folgenden Java-Quelltext. Welche Ausgaben erzeugt das Programm? Erläutern Sie den Programmlauf.

```
public class StatClass
{
 static int i;
 public static void statMethod ()
 {
   i=i*2;
 }
}
public class statMain {
 public static void main(String[] args)
 {
  StatClass r = new StatClass();
  StatClass s = new StatClass();
  r.i=10;
  StatClass.i=10;
  StatClass.statMethod();
  r.statMethod();
  s.statMethod();
  System.out.println(StatClass.i);
  System.out.println(s.i + " " + r.i);
 }
}
```

### Aufgabe 4: Bruchrechnen

4.1 Definieren Sie eine Klasse *Bruch* mit den Eigenschaften *Zähler*, *Nenner* und den Methoden *bruchEingeben()* und *bruchAusgeben()*. Außerdem soll ein Konstruktor definiert werden, der den Bruch mit 0/1 initialisiert. Schreiben Sie ein Testprogramm für die Klasse *Bruch.*

4.2 Eine Klasse kann mehrere Konstruktoren haben (Überladung). Anhand der Parameterliste wird bei der Instanzierung entschieden, welcher Konstruktor aufgerufen wird. Die Konstruktoren der Klasse *Bruch* (vgl. Aufgabe 4.1) sollen überladen werden, damit folgende Objektinstanzierungen möglich werden:
*Bruch b1=new Bruch();// Zähler wird mit 0 und Nenner mit 1 initialisiert*
*Bruch b2=new Bruch(1,4);// Zähler und Nenner wird mit bel. ganzen Zahlen initialisiert*
*Bruch b4=new Bruch(4); // Gesamtbruch b4 wird mit Wert (hier 4) initialisiert*

4.3 Erweitern Sie die Klasse *Bruch* durch Methoden zur Addition, Subtraktion, Multiplikation und Division von zwei Brüchen.
Beispiel:
*b3=b1.add(b2 ;)*
*b4=b1.sub(b2) ;*
*etc.*

4.4 Zeichnen Sie ein UML-Klassendiagramm.

4.5 Entwickeln Sie ein Testprogramm für die Klasse *Bruch.* (evtl. mit einem Menü (+,-*,/)

### Aufgabe 5: Zeitverwaltung

5.1 Erstellen Sie zur Bearbeitung von Messwerten einer Stoppuhr eine Klasse *Time* mit ganzzahligen Eigenschaften für die Stunden, Minuten und Sekunden. Die Objekte sollen durch Konstruktoren entweder mit 0 oder mit übergebenen Startwerten initialisiert werden.

5.2 Schreiben Sie eine Methode zur Ausgabe der Werte im Format <h:m:s>.

5.3 Erweitern Sie die Klasse *Time* durch Methoden zur Addition und Subtraktion von 2 Messzeiten.

5.4 Entwickeln Sie ein Testprogramm für die Klasse *Time* und zeichnen Sie das UML-Klassendiagramm.